LIBRO BLANCO | MARZO DE 2016

DevOps y los probadores

Paul Gerrard, un gurú de las pruebas y consultor, analiza en una serie de artículos una amplia gama de cuestiones relacionadas con las pruebas. En esta ocasión, habla sobre la adopción de DevOps desde el punto de vista de los probadores y las pruebas. DevOps forma parte de un enfoque general que las organizaciones utilizan para entregar *software* de gran calidad con frecuencia. El resultado más evidente de las implementaciones exitosas de DevOps es la reducción del tiempo que tardan los cambios en el *software* en pasar de ser una idea a la fase de producción.

Paul Gerrard
Gerrard Consulting

Con el patrocinio de

ca technologies

¿Qué significa DevOps para los probadores?

Contexto

En este artículo, deseo hablar sobre la adopción de DevOps desde el punto de vista de los probadores y las pruebas. El movimiento (a falta de una etiqueta más adecuada) de DevOps está progresando rápidamente. Al igual que muchas otras iniciativas del sector, su adopción ha sido más rápida que la propia definición del movimiento. El concepto de DevOps aún no está bien definido y los matices de la cultura, la capacidad emergente de las nuevas tecnologías y la gran variedad de casos prácticos (en su mayoría exitosos) hacen que las cuestiones que nos conciernen sigan siendo objeto de amplios debates.[1]

En función de con quién hable, DevOps puede ser una solución para un problema o un objetivo en sí mismo. En algunas empresas, el objetivo es "hacer la transformación digital", y DevOps forma parte del enfoque general para realizar entregas de gran calidad con frecuencia. En este artículo, partiremos de este contexto. Ahora bien, para la comercialización de tecnologías y servicios relacionados con DevOps, este objetivo puede verse difuminado. Se suele subestimar el reto del cambio cultural (o, más en concreto, el cambio de comportamiento) necesario para el éxito.

El otro supuesto en el que me baso es que DevOps es un concepto completamente nuevo para los probadores que intervienen en él y a los que les afecta este movimiento. He diseñado este artículo como una introducción a DevOps para este tipo de probadores y como un análisis de su impacto en las prácticas de las pruebas. En el caso de los profesionales de DevOps experimentados, espero que este artículo les resulte de utilidad. Los profesionales no probadores podrán al menos ver DevOps desde el punto de vista de los probadores.

Para los no iniciados: ¿qué es DevOps?

Para simplificar, DevOps es una etiqueta que se aplica a la noción de que los equipos de desarrollo y de operaciones de sistemas deben colaborar de forma más estrecha. En lo que se conoce como el "canal de entrega", desde la confirmación del código fuente hasta las operaciones de producción, los desarrolladores adaptan y automatizan algunas de las actividades de las operaciones. Los equipos de operaciones tienen una mayor visibilidad de las actividades de los desarrolladores y un cierto grado de influencia sobre ellas. El principal motivo de todo esto es la aceleración del despliegue y la implementación del *software*. Al reunir a los equipos de operaciones (Ops) y de desarrollo (Dev) para que trabajen juntos (de forma eficaz en un equipo ágil), se implementan lo que podríamos llamar "operaciones ágiles".

El resultado más evidente de las implementaciones exitosas de DevOps es la reducción del tiempo que tardan los cambios en el *software* en pasar de ser una idea a las operaciones de producción. Cuando un desarrollador dice que se ha "hecho" un cambio en el *software*, la transición al uso en la fase de producción se realiza con la ayuda de una automatización generalizada. Las herramientas y los procesos de automatización se utilizan en la configuración de sistemas, el proceso de creación, las pruebas, el paso de la implementación a las pruebas, los entornos de preproducción y de producción, la monitorización posterior a la implementación, la evaluación y la operación.

¿Debemos entender entonces que DevOps es solo una cuestión de herramientas?

En uno de sus niveles, el objetivo de DevOps es eliminar los cuellos de botella del canal de entrega mediante la automatización. Sin embargo, la automatización de procesos dispuestos en etapas continúa requiriendo gobernanza. La mayoría de los procesos automatizados no son realmente autónomos, ya que no pueden realizar sus tareas sin intervención humana para las tareas de mantenimiento y la gestión de excepciones. Un proceso de DevOps completamente automatizado carece de sentido si no se tiene en cuenta el factor humano. Aunque las herramientas realizan una gran cantidad de tareas complicadas, son los profesionales que ejecutan el proceso los que hacen que este funcione (o que no lo haga).

Entonces, ¿debemos entender que DevOps consiste tan solo en que los equipos de desarrollo y de operaciones colaboren más estrechamente con la ayuda de las herramientas?

No, tampoco es eso. Las entregas entre procesos automatizados suelen implicar otros procesos (normalmente, pruebas de algún tipo). Las pruebas automatizadas las deben crear los desarrolladores y los probadores. Los resultados de estas pruebas se centran en aportar información suficiente para que otros procesos (o, casi con la misma frecuencia, otras personas) puedan realizar la transición entre las etapas del canal. Los probadores y desarrolladores que realizan pruebas controlan que el proceso de DevOps realice entregas de forma correcta y fiable.

Más de uno podría decir: “Me duele la cabeza, ¿qué es DevOps exactamente?”. Tengo que decir que se trata de una disciplina emergente en constante cambio. Esta cuestión se plantea y analiza con profundidad en una excelente publicación aquí.[1] Este debate tuvo lugar tan solo unas cuantas semanas antes de que escribiera este artículo. Como se puede ver, todavía no disponemos de una definición asentada de DevOps. Y quizás nunca la tengamos.

¿Qué implicaciones tiene esto para los probadores? Implica que todavía no existe “el único método adecuado” y que aún no se ha fijado el papel que deberá desempeñar usted en un régimen de DevOps en evolución (y todos los regímenes evolucionan). Para ello, estas son las dos contribuciones más importantes que debe hacer:

1. Debe prestar atención a las tareas engorrosas y trabajar para que resulten menos penosas.
2. Debe identificar las oportunidades e intervenciones que agregarán valor al proceso de DevOps.

Si tuviera que citar un mantra que describiera de la forma más acertada qué es lo que impulsa la transición a DevOps, diría que es “si es un engorro, hágalo más”. Puede parecer un tópico, pero lo utilizaré como contexto para la implementación y mejora de las prácticas de pruebas de DevOps.

Si es un engorro, hágalo más (a menudo)

La dificultad o el engorro que sentimos al realizar una tarea concreta nos afecta negativamente. Si no queremos hacer una tarea, tendemos a posponerla. Cuando por fin nos ponemos manos a la obra, la tarea nos resulta aún más molesta. Esto se aplica a las visitas al dentista, a limpiar el garaje, a integrar *software*, a las pruebas, etc. Según nuestra experiencia, estas tareas suelen ser más traumáticas cuanto menor sea la frecuencia con las que las hagamos. Martin Fowler indica tres motivos por los que ejecutar determinadas tareas de forma frecuente o incluso continua hace que resulten menos engorrosas.[2]

El primero es que las tareas más largas y complejas cuestan de planificar, gestionar y controlar, por lo que dividirlas en otras más pequeñas hace que resulten más sencillas, menos arriesgadas y, si algo sale mal, más fáciles de revertir. El segundo es que muchas tareas (y las pruebas son un ejemplo perfecto) aportan retroalimentación. Si se recibe dicha retroalimentación con frecuencia y prontitud, los problemas se pueden solucionar con rapidez y seguridad antes de que se pierda más tiempo. El tercer motivo es que si realizamos una actividad con mayor frecuencia, la acabaremos haciendo mejor. Además, aprenderemos a hacerla de forma eficiente. Asimismo, es posible que veamos oportunidades de automatizarlas de algún modo.

Desde el punto de vista del probador, este mantra nos obliga a tomarnos mucho más en serio la noción de la automatización en el proceso de las pruebas. Si hay que intervenir manualmente (lo habitual es que sea entre las etapas automatizadas del proceso de DevOps) estas intervenciones se considerarán los puntos engorrosos: los cuellos de botella, las causas de retrasos y los aspectos del proceso propensos a errores y potencialmente menos fiables. Las pruebas manuales son un engorro. Vale, puede que le encante realizar pruebas exploratorias; puede que tenga miedo de que solo un ser humano como usted pueda detectar los molestos errores que la automatización nunca detectará; y que piense que un probador como usted sea la única persona de confianza para impedir que se produzcan desastres.

Como probador, puede resultarle complicado confiar en que los desarrolladores y la automatización realicen las tareas de pruebas adecuadamente. Si es un engorro, debe hacerlo más a menudo.

Pruebas, automatización y confianza

Se ha debatido mucho sobre el significado de, por ejemplo, las comprobaciones y las pruebas,[3] así como sobre la fiabilidad que podemos otorgar a los probadores, las comprobaciones y la automatización.[4, 5]

No digo que podamos confiar ciegamente en las comprobaciones automatizadas. Sin duda, debemos ser más sofisticados. No obstante, para los propósitos del presente artículo, sí que podemos al menos descomponer las pruebas y la actividad de ejecución de pruebas en cuatro componentes.

1. Comprobaciones que los desarrolladores pueden automatizar dentro de sus procesos de integración continua y registro de componentes
2. Comprobaciones que se pueden automatizar (lo suelen hacer los probadores de sistemas) para ejecutar transacciones en el ámbito de API, de vinculación o integrales
3. Pruebas que pueden realizar comprobaciones de compatibilidad para poner en práctica la compatibilidad entre navegadores, sistemas operativos y plataformas
4. Pruebas que solo puede hacer un ser humano

En este artículo solo puedo aportar unas cuantas recomendaciones sobre cómo realizar tales distinciones (evidentemente, cada entorno es distinto). La pregunta más pertinente para este artículo es la siguiente: "¿Cómo puede «librarse» un probador de las comprobaciones manuales finales?". Ya he hablado con anterioridad sobre la eliminación de las comprobaciones manuales finales.[6] Se precisa un esfuerzo y una confianza proactivos.

A continuación, indico las principales áreas en las que debe concentrar sus esfuerzos:

1. Siempre que sea posible, las comprobaciones manuales que se puedan realizar en el nivel de los componentes se deben remitir a los desarrolladores. Como probador, puede sugerir pruebas de este tipo en una sesión de puesta en común o una presentación colaborativa. Puede que tenga que escribirlas usted mismo e incluirlas en el régimen de integración continua.
2. Puede que las pruebas integrales de la interfaz de usuario requieran automatización. Este tipo de pruebas se deben reducir a la mínima expresión, ya que tienden a ser tediosas, inestables y requerir mantenimiento con frecuencia. Plantéese si se deben ejecutar cada vez que registre código o si se podrían reservar para su uso exclusivo en lanzamientos de mayor envergadura y menos habituales.
3. ¿Qué pruebas que solo se pueden realizar manualmente se podrían ejecutar en los componentes que aún no se hayan integrado en un candidato de lanzamiento? ¿Se pueden realizar las pruebas manuales en las sesiones de puesta en común con los desarrolladores? ¿Hay alternativas a estas pruebas? ¿Podrían ser útiles la realización de guiones gráficos o la creación de prototipos siguiendo el estilo del desarrollo basado en el comportamiento? ¿Podrían las comprobaciones de la interfaz de usuario efectuarse en modelos o *wireframes*?
4. ¿Cuáles son las comprobaciones que se deben ejecutar solo una vez manualmente, en contraste con las comprobaciones que se deben mantener para fines de regresión y son candidatas para la automatización?

Anteriormente, he mencionado la noción de la confianza. Otra forma de examinar esta cuestión es especular cómo se podría probar con fiabilidad un sistema si no se realizó ninguna prueba manual final. Imagine un entorno en el que las herramientas realizaran todas las pruebas. ¿Estarían sus preocupaciones dominadas por el hecho de que, simple y llanamente, no se fía de que los desarrolladores hayan hecho un buen trabajo de pruebas? Si adelanta la planificación de las pruebas (como recomendé en mi artículo anterior), esas dudas deberían reducirse. Si, como probador, actúa más como un explorador para identificar los riesgos, evaluarlos, seleccionar pruebas y garantizar que estas se incorporen al desarrollo y la automatización, sus preocupaciones podrían minimizarse.

Es indudable que debe dejar de creerse el baluarte de la calidad, la última línea de defensa y la única persona que se preocupa. Debe pensar más como un visionario, un identificador y gestor de riesgos, un explorador, un vehiculador y un instructor o mentor.

Práctica, monitorización y mejora

Con todas las buenas intenciones de reducir o eliminar la dependencia de las comprobaciones manuales finales, aún pueden colarse errores. Cuando el *software* se lanza a la etapa de producción, surgen los problemas. Una de las disciplinas más importantes de DevOps desde el punto de vista de las operaciones es un grado de monitorización más profundo.

Se debe monitorizar en todas las capas, desde los componentes y las simples transacciones de las aplicaciones, hasta la integración y los mensajes y, por supuesto, la propia infraestructura. Uno de los objetivos de la monitorización es generar alertas sobre los fallos antes de que los usuarios noten sus consecuencias. Es un objetivo bastante ambicioso, pero es el definitivo.

Cuando surgen problemas en la fase de producción, la tarea que se debe hacer es utilizar los análisis derivados de la monitorización para no solo rastrear su causa y resolverlos, pero también para perfeccionar el proceso de prueba (ya sea automatizado o manual) y reducir la probabilidad de que se produzcan problemas semejantes en el futuro. El papel de las pruebas y los análisis en todo el proceso del canal se planteó y se debatió en este artículo.[7]

A las pruebas automatizadas del proceso de DevOps se les podría llamar "monitorización". Además de la monitorización en la fase de producción, se podría afirmar que la monitorización a lo largo de todo el proceso de DevOps y encaminada a la fase de producción amplía el alcance de las pruebas. Por lo tanto, DevOps no reduce las funciones de los probadores.

Conclusión

Hace poco alguien me preguntó: "¿Cuándo no debería intentarse la implementación de DevOps en una organización?". Es una buena pregunta, pero creo que lo que subyace es la duda de si DevOps ha venido para quedarse y si los probadores deberían tomar buena nota de ello. Mi respuesta es sencilla.

¿Por qué no sería deseable que el personal de desarrollo y el de operaciones dialogaran? ¿Por qué no sería deseable disponer de compilaciones e implementaciones más fiables en las fases de pruebas y de producción? ¿Por qué no sería deseable disponer de la mejor tecnología para respaldar canales más precisos, eficientes e informativos? DevOps es algo bueno, pero no siempre es fácil conseguirlo. No hace falta decir que requiere un cambio cultural, y eso no siempre es fácil.

DevOps nos aporta a los probadores una mayor influencia en las fases tempranas de los proyectos, nos obliga a plantearnos más seriamente la automatización en las pruebas, el aprovisionamiento de información y la toma de decisiones. Los probadores deben adoptar DevOps porque les ofrece la oportunidad de ser proactivos y ganar una autoridad y un respecto mayores en los equipos de proyectos.

Acerca del autor

Paul Gerrard es consultor, profesor, escritor, *webmaster*, desarrollador, probador, conferenciante, entrenador de remo y editor. Ha realizado tareas de consultoría en todos los aspectos de las pruebas del *software* y el control de calidad, y se ha especializado en el control de las pruebas. Ha presentado tutoriales y charlas destacados en congresos sobre pruebas en toda Europa, EE. UU., Australia, Sudáfrica y, en ocasiones, ha sido galardonado por ello.

Formado en la Universidad de Oxford y el Imperial College de Londres, Paul ganó en 2010 el premio Eurostar European Testing Excellence Award y, en 2013, fue galardonado con el European Software Testing Award (TESTA) en reconocimiento de su carrera profesional.

En 2002, Paul escribió el libro *Risk-Based E-Business Testing* con Neil Thompson. Posteriormente, escribió *The Tester's Pocketbook* en 2009. Paul fue coautor del libro *The Business Story Pocketbook* con Susan Windsor en 2011 y escribió *Lean Python* en 2014.

En 2014, Paul fue el Director de Programa del congreso EuroSTAR Conference que tuvo lugar en Dublín.

Es el Director de Gerrard Consulting Limited, de TestOpera Limited y anfitrión del Test Management Forum.

Correo electrónico: paul@gerrardconsulting.com
Twitter: @paul_gerrard
Página web: gerrardconsulting.com

Para obtener más información, visite **Desarrollo y pruebas** con CA Technologies.

Comuníquese con CA Technologies en ca.com/es

CA Technologies (NASDAQ: CA) crea *software* que impulsa la transformación de las empresas y les permite aprovechar las oportunidades que brinda la economía de las aplicaciones. El *software* se encuentra en el corazón de cada empresa, sea cual sea su sector. Desde la planificación hasta la gestión y la seguridad, pasando por el desarrollo, CA trabaja con empresas de todo el mundo para cambiar la forma en que vivimos, realizamos transacciones y nos comunicamos, ya sea a través de la nube pública, la nube privada, plataformas móviles, entornos de *mainframe* o entornos distribuidos. Para obtener más información, visite **ca.com/es**.

Referencias

1. "What is DevOps", The Agile Admin, http://theagileadmin.com/what-is-devops/
2. "Frequency Reduces Difficulty", Martin Fowler, http://martinfowler.com/bliki/FrequencyReducesDifficulty.html
3. "Testing and Checking Refined", James Bach, Michael Bolton, http://www.satisfice.com/blog/archives/856
4. "A New Model for Testing", Paul Gerrard, http://dev.sp.qa/download/newModel
5. "The New Model and Testing v Checking", Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/659
6. "How to Eliminate Manual Feature Checking", seminario web de Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/622
7. "Thinking Big: Introducting Test Analytics", Paul Gerrard, http://blog.gerrardconsulting.com/?q=node/630

CS200-181159_0316